Advancements in Energy-Efficient High-Performance Architectures for Edge Computing: Insights from Pipestitch, RipTide, and REVAMP

Nomita Meka
Department of Computer Science
University of North Texas
Denton, USA
nomitameka@my.unt.edu

Venkata Kishan Kapuganti
Department of Computer Science
University of North Texas
Denton, USA
VenkataKishanKapuganti@my.unt.edu

Akshaya Yalla
Department of Computer Science
University of North Texas
Denton, USA
akshayayalla@my.unt.edu

***Abstract*—This survey paper is going to constitute of three major potential contributions in the field of data flow architectures related to energy efficient, Specifically they are : Pipestitch, RipTide, and Heterogeneous CGRA Realization. The Pipestitch is an architecture which is primarily designed to enhance the performance in the sparse workloads and at the same time the goal of reducing the consumption of energy which makes the system as energy efficient. This Pipestitch generally introduces a lightweight hardware threads, which is generally a new programming model and it can also be stated as a new synchronization network, which is making some notable advancements in the performance improvements compared to other existing architectures like RipTide that has the minimal amount of increase in the area and the consumption of energy. So as we speak about the RipTide, it is generally termed and stated for the importance on the energy efficiency and the usage of the dataflow compiler and also it mainly uses the architecture which is responsible for the offloading of the computational kernels on to the CGRA fabric. Even though as stated the RipTide efficiency, this one is struggling to cope up with the performance in the sparse applications. Then here comes the one more approach in the survey report which is the integration of the some numerous processing elements into the CGRA framewrok to make the balance of the flexibility and the performance and then making it the energy efficient which makes the system suitable for the complex and sparse computations. So this survey as stated focuses on the methods, results and the implications that has been noted from these architectures which also offering the current advancements and the further future research innovations in the concept of the energy efficient dataflow computing. Pipestitch employs a new programming model as discussed earlier with the introduction of control-flow operator, and synchronization network to achieve a 3.49x performance increase over the state-of-the-art RipTide, with minimal area and energy overheads.**

I. Introduction

As we have seen there is sudden increase in the advancements in the technology in the recent years, which has made an imporatant requirement for the emergence of sophisticated sensing and computing applications. As these applications are increasingly deployed at the main edges of the network, which are in other words can be stated as the networks that are beyond the reach of the conventional power and the infrastructure. [1] The example that are real life applications linked to this area are : infrastructure monitoring, wearable health devices, and satellite systems. so this kind of systems generally needed the large volumes of data for the local processing and also there is a need for the efficiency and flexible architectures for computing. [3]

As we all aware of the fact that, the general application usually faces the notable challenges in the extreme edge computing, which are majorly due to the minimal energy constraints and at the same time there is a need for the high performance. So conventional architectures are not capable of pulling the goal , which are often wasting some numerous amount of energy on the data movement, instruction fetch, and pipeline control. [5] So all this lead the researchers to shift the focus on to development of the energy minimal data flow architectures which are having the advanced performance and the same time they are highly energy efficient and offers programmability.

So this survey paper going to make a focus on the notable contributions that made by three research papers that focuses on the energy-minimal dataflow architectures: Pipestitch, RipTide, and Heterogeneous CGRA Realization. Where each of them addresses the challenges of the extreme edge computing in distinct ways, which are providing the notable insights into the current state of the research and the potential future directions. [2]

Pipestitch is generally considered as one of the prominent research in the field of computing systems, which is a energy-minimal CGRA architecture, here CGRA is Coarse-Grained Reconfigurable Array, which is specifically designed to enhance the performance in the sparse work loads. Generally it begis with the introduction of the light weight hardware threads and a synchronization network called the SyncPlane, which is allowing for the efficient parallelism. This architecture makes the prominent move in the performance of the computing systems with very minimum area increase and energy overhead, which is making this as a promising extreme

edge applications. [1]
RipTide is also one of the key architecture, that has been making insights on the energy efficiency by the method of removing the entire computational kernels onto the CGRA fabric. Where this method of approach making the significant improvement in the energy minimizing, which is making the RipTide highly efficient. But as there are positive impacts and like wise the negative impacts, which include challenges in the maintaining of the high performance for the sparse applications, which making the systems limited in some scenarios. Now the exploration of Heterogeneous CGRA proposes the method of integration of the processing elements into the CGRA framework. where the ultimate goal of this is to balance the flexibility, performance, and energy efficiency of the systems that is making it suitable for a wide range of applications, including the ones that involve complex and sparse computations. As we stated by the mixture of different types of the processing elements, this heterogenous CGRA offers a more adaptable and versatile solution to the varying demands of the extreme edge applications.
The main area of goal of this survey is to analyze the importance and the weaknesses of the proposed architectures, thereby discussing about the future implications or the research that can be made on this data flow. which is energy minimal data flow computing. [6]

II. Background

As there is a steep increase in the sensing technologies, and there is a specific need for the real time data processing, which made the potential interest in the field of extreme edge computing. These systems must operate under strict energy constraints and at the same time they should be responsible in the delivering of high-performance computing capabilities to process the data they collect locally. These extreme edge computing has been raising many challenges, thouugh we have the conventional architectures but they are very much inefficient for this usecase as they waste more energy, to be precise this traditional architecture consume more energy in various steps of the flow of the execution that includes data movement, instruction fetch, and pipeline control. [10] So to address the challenges being faced by the traditional ones, the researchers have come up with an energy minimal data flow architectures, where these are designed in such a way, they reduced the consumption of energy and at the same time maintaining the performance of the complex workloads, where the term comes into picture extreme edge computing scenarios. So this proposed architectures have achieved this with the spatial distributing computation across a grid of processing elements (PEs), which execute operations and communicate intermediate values via a network-on-chip (NoC). So generally this distribution try to minimize the overhead that is being linked to the data movement and the control operations. which are the main ways of consumption of more energy which are being removed in the proposed way of data flow. [11]
As through this survey we made some notices that though the previous work has made considerable progress in the improvement of the energy efficiency. For now if we consider RipTide, it is a architecture that has chosen a way of offloading the entire computational kernels onto the CGRA fabric. Which made the progress of the elimination of the need for the vast amount of data movement and the instruction fetch operations. [5] RipTide has made a notable amount of reduction in the consumption of energy. But the performance of the systems has been the challenge in this approach. Mainly in the systems that are prone towards the sparse applications or the sparse computations, that are very much common in the Machine learning and the digital signal processing. These applications require architectures that can efficiently handle irregular data patterns and exploit parallelism to improve performance. So there is a very much clear statement that there is a need for the development of the architecture that is not only focusing on the reduction of energy consumption or in other words efficient in energy but also need to have the enhancement in the performance on the sparse computations. [3]
So PipeStitch is one such architecture that lead to the development, which introduces a light weight thread and a novel synchronization network which allows the parallel execution of the operations within the nested loops.

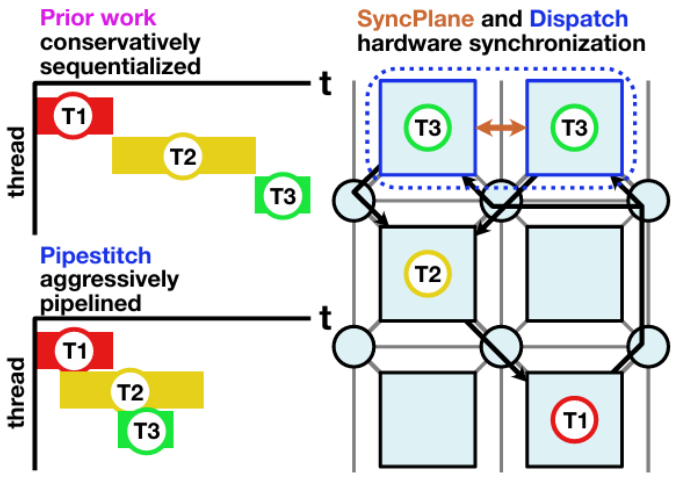

Fig. 1. Comparison of Without PipeStitch and with PipeStitch

From the Fig:1, we can clearly see that, there is an illustration of the comparison between the traditional and the PipeStitch approaches for the handling of the threads in the energy minimal CGRAs, as We can see the left side of the image is about the Conservative sequentialization which is the Thread execution of the Sequential manner, where Each thread wait for the previous one to get complete, which leads to the longer execution times and also the under utilization of the resources.
In the left side we also see the PipeStitch architecture which is following the Aggressive pipelining, where the threads are executed in an pipelined fashion where we can also term it as overlapped fashion, this allows the simultaneous execution of the threads and this leads to the better utilization of the

resources and the reduced level of execution times. [1]
In the right side of the picture if we see thetre is the SyncPlane and Dispatch Hardware Synchronization, so generally SyncPlane is the one which make sures that threads are coordinated correctly during their overlapped execution. and Dispatched Hardware is the one which manages the dispatching of the threads to the various processing units. So with the image we can state that PipeStitch generally adds the architectural support and microarchitectural support for the pipelining of the light weight threads on the energy minimal CGRAs.

III. Motivation

The motivation that runs behind this innovations are from the needs that has been noticed, which are systematically reviewed and compared with the most recent advancements that has been made in the energy minimal dataflow architectures. By making the analysis of the major key contributions from Pipestitch, RipTide, and Heterogeneous CGRA Realization, this survey paper aims to make a notable highlights on the current state of research and also helps in the identification of the potential directions for the future research and also Understanding the strengths and limitations of each approach will give us valuable insights and patterns that needs to be followed for researchers and practitioners working to overcome the challenges of extreme-edge computing. [1]

IV. Methodology

In this section of the survey report, we will be providing the complete explanation of the methodology and the algorithms used in the selected papers: Pipestitch, RipTide, and Heterogeneous CGRA Realization. We have also noted the implementation steps and then at the end provided the comparative analysis. which mainly notices the strengths and the weaknesses of the respective approaches that have been proposed by the researchers.

A. Pipestitch: An Energy-Minimal Dataflow Architecture with Lightweight Threads

As discussed, this Pipestitch is a most advanced type of architecture, that has energy-minimal CGRA architecture, which is specifically designed to be as a solution for the challenges that has been faced by the traditional architectures, which are having some bottelnecks in terms of the performance in the sparse workloads. So this Pipestitch will make sure it has energy efficient while maintaing the performance of the computing systems throught he sparse workloads or in other words in a environment of extreme edge computing. The main core implementation of the pipestitch lies in the introduction of the light weight hardware threads and the SyncPlane sychronization network. So generally this programming model is streamlined through the foreach construct, which is enabling the approach of the specification in the parallelism within the nested loops easily. So the compiler is initially translating these constructs into a light weight threads which run concurrently on a CGRA fabric, then they are made to be synchronized by the help of dispatch operators and the SyncPlane. [1]

So as we speak about the modifications that made to the arhcitecture , it includes the new dispatch gates and a synchronization mechanism, which is making sure the fact of the ordered execution of the threads. so this proposed design has made a significant achievement by gaining 3.49x times in the performanc eimprovement over the RipTide with only 1.10x increase in the area which is very minimal area and a 1.05x increase in the energy consumption. So this metrics are the evident of the Piepstitich working power compared to other architectures. So as we speak about the strengths linked to this approach, it has the notable performance improvement and efficient thread synchronization and a simplified programming model for the parallel excecution. [4]
Now as we move to implementation section of this particular approach.

- **foreach construct:** here it includes the writing of a C program that constitutes of foreach construct.
- **Compilation:** Now in this step the compiler translates the given program into a DGF graph, which is data flow graph with the light weight threads.
- **Mapping of the DFG to CGRA fabric:** In this step the compiler works on the mapping of the operations onto a processing elements which are PEs in short and it also configures the networks on chip which is NoC in short.
- **Insertion of the dispatch operators:** So at this stage the implementation focuses on the addition of the dispatch operators which are responsible for the thread synchronization and it also make sures the ordered execution.
- **Execution:** Now in this final step the CGRA fabric performs the execution of the DFG, with all the threads pipelines and synchronized via the SyncPlane.

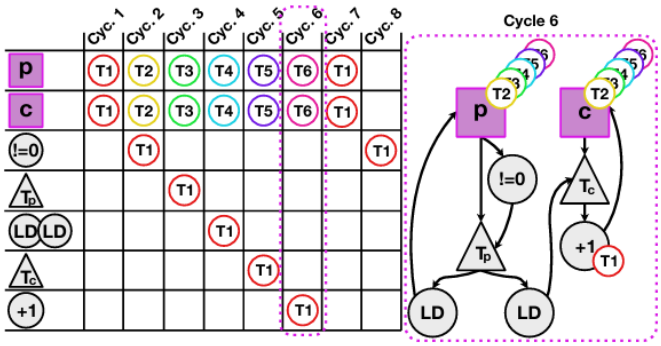

Fig. 2. Without PipeStitch

In the above fig:2, It demonstartes the execution of the threads on a data flow graph(DFG), as in the figure we can see the each row represents a thread like T1, T2, .. and each column denotes a cycle, the purpose of the labelled boxes shows the P and C pipeline and the control operations, then the diagram on the right side of the image shows the state of the system, which highlights the blocking nature of the traditional approaches.

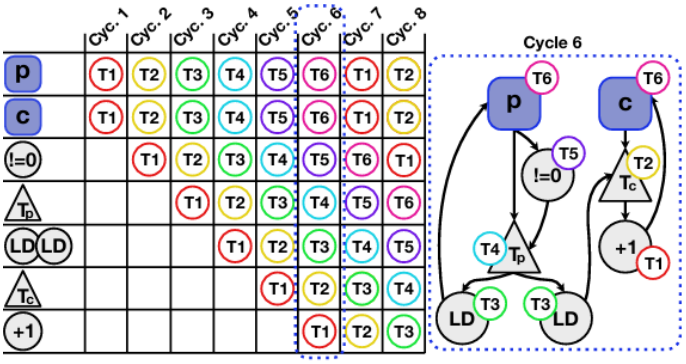

Fig. 3. with PipeStitch

The above Fig:3, is showcasing the thread executin with the usage of the PipeStitch architecture, where all the threads are unblocled and pipelined through the inner loop. where here it is clear that multiple threrads can execute at same time , which improves the utilization of the resources and the enhancement in the performance. The aggressive pipelining allows threads to progress without waiting for the previous thread to complete.

As we summarize both the above Fig:1 and Fig:2. Threads are blocked at the head of the inner loop , which leads to low utilization and the drop in the performance as the threads wait for their turn to come for the execution. Now with the usage of the PipeSitch architecture, the threads are unblocked and they are throught the inner loop which avoid the blockage leads to the more utilization of the resources and the enhancement of the performance due to the fact that it offered concurrent execution of the independent threads. [15]

B. RipTide: A Programmable Energy-Minimal Dataflow Compiler and Architecture

In this proposed approach , it takes the steps of removing or offloading all the computational kernels onto a CGRA fabric. so this approach mainly focuses on the reduction of the energy consumption as this strategy minimizes the energy wasted in the traditional CPU operations with the help of removing the need for the high level of movement of the data and also the instruction fetch linked to it. Generally the RipTode compiler works in a arbitary control flow and also has the memory access and it also involves in the mapping of the high level C code to the CGRA fabric while at the same time managing the order of the tokens through the diverging paths, loops and memory accesses.

Now it begins with the architectural improvements based out in this approach has the mapping of one operation to the each PE, which is there to enhance the switching activity, it also available in the disallowance of the reordering to avoid the tagging values and also helps in the elimination of the buffers in the Noc. [7] So all this measures put up a hand or level by level in the enhancement of the energy efficiency. Even though it has the high energy efficiency, this approach is struggling a lot with the issues related to the performance in the sparse applications which are in the extrene edge computing, which is beacause of the fact of they don't have the ability to cope up with the parallelism effectively. The major thing here is that, they are high energy efficient and a simplified architecture, but the limitation is related to the performance and it also falls under less flexibility in the handling of the irregular patterns in the data and the work flows which are complex in nature.

Now we write the short report on the steps involved in the implementation of this methodology.

- **C program:** In this step we write a high level code in the C programming language.
- **Compilation:** The compiler works on this step where it translates the program into the operations mapped to the CGRA fabric
- **Mapping of the operations to PEs:** Here in this stage of the implementation, Each of the operations are generally mapped to the PE, to help in the minimization of the Switching and the energy consumption.
- **Configuration of Noc routers:** Here at this stage, Noc routers are used for the control flow and also buffers are made to be eliminated to reduce the consumption of the energy.
- **Execution:** Here at this final stage of the implementation the CGRA fabric will take care of the execution with the minimal energy wastage.

C. Heterogeneous CGRA Realization

This approach speaks about the exploration of the integration of many diverse processing elements within the CGRA framework to balance the flexibility, performance and the energy efficiency. so this type of approach generally makes the architecture suitable for the sparse computations, [12] which is offering a more adaptable nature for the changing needs of the extreme edge computing. In this approach the CGRA fabric includes the different kind of processing elements, which is making it to a suitable kind for the wide range of the applications with the performance and energy efficiency enhancements and the requirements associated with them.

The compiler for this Heterogeneous CGRA Realization is designed in such a way that, it map the operations to the appropriate PEs, which ensures the efficient utilization of the heterogenous resources. As this architecture has the ability to provide a high flexibility and the adaptability with the balanced performance and the energy saving, [13] and it is also capable of handling the dynamically changing requirements of the applications.

Below are the steps to be followed for the implementation of this approach.

- **Designing of the CGRA fabirc:** As this fabric is designed in such a way that it is heterogenous in nature, which includes diverse types of PEs to handle a wide variety of applications.
- **Compiler development:** Here it make sures to map the various applications to the appropriate PEs in the CGRA.

- **Program implementation:** Here at this stage it focuses on the compilation of the program into operation which are mapped to the CGRA fabric.
- **Mapping:** The compiler generally finds the best appropriate PEs to the respective operations and tries to optimize the performance and the energy efficiency.
- **Configure the Noc:** Here we make sures the fact of effective communication between the heterogenous PEs.
- **Execution:** The CGRA fabric executes the operations, by the inclusion of the diverse PEs to balance the performance and energy efficiency associated to it.

As we summarize the survey report related to the methodologies, the Pipestitch is the most significant approach that has performance improvements and the efficient thread synchronization, but it introduces a complexity in the CGRA design and also offers potential compiler optimization problems. whereas the RipTide is to good in terms of the energy efficiency through the approach of the efficient operation mapping and a simplified architecture but has problems related to the performance for the sparse applications. [16] The Heterogeneous CGRA Realization offers a high flexibility and adaptability to many diverse applications and balanced performance, and energy efficiency, but the concerns here is that it had increased design complexity and potential optimization challenges.

V. Evaluation

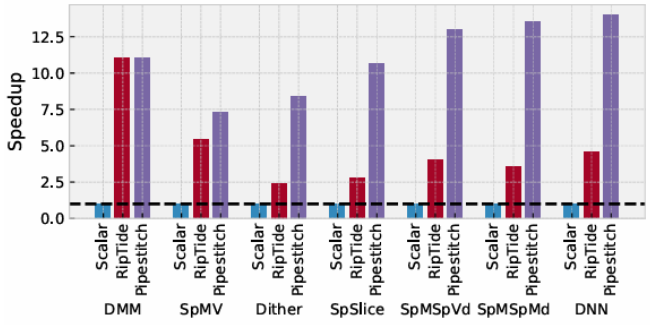

Fig. 4. Speedup compared to scalar for RipTide and Pipestitch.

From the Fig:4, we can state that **Pipestitch** has been the best performing architecture so far from the survey with the notable performance enhancements and the efficient way of thread synchronization is one more thing to be considered and it also offers a simplified way of programming model. As there are dark side of the approach which includes the complexity associated with the CGRA design and there are some potential challenges linked to the optimization of the compiler. [9]

RipTide was predominantly best architecture in terms of the energy efficiency and the architecture is simple in nature, which is effectively responsible for the regular control flow and the memory access. But as we discussed the performance is the issue related to the Riptide, which is one of the drawbacks when working on the sparse applications and it also have complex control flows.

Heterogeneous CGRA Realization offers very high flexibility and the adaptability when it matters the most and it is the go to approach in the balancing of the performance and energy efficiency, which is optimally suitable for the compelx workloads which are dynamic in nature. [11] The challenges linked to this approach are design complexity.

Factors	Pipestitch	RipTide	Heterogeneous CGRA Realization
Performance	High	Limited	Balanced
Energy Efficiency	Moderate	High	Balanced
Flexibility	Low	Low	High
Design Complexity	High	Low	High
Compiler Optimization	Challenging	Simplified	Challenging

TABLE I
COMPARISON OF PIPESTITCH, RIPTIDE, AND HETEROGENEOUS CGRA REALIZATION ARCHITECTURES

The above Table:1, is the quick grasp kind of representation of the evaluation of the approaches that has been surveyed throughout this report on some of the factors that are important for an architecture to be competent in the field of computer architecture.

VI. Conclusion

We can conclude that the future research in the minimal energy dataflow computing can be done by focusing on few of the several key areas which will be improving the capabilities and the efficiency of the CGRA architectures. While the advanced complier optimization is essential for the fully exploiting the potential of Pipestitch, RipTide and Heterogeneous CGRA Realization means that which is done by improving the thread management, parallelism and the resource allocation. The real-time adjustments towards the workload and environmental changes, enhancing the performance and energy efficiency can be done by enabling the dynamic reconfiguration and adaptation mechanisms. [3] As by integrating with the emerging technologies such as non-volatile memory and advanced interconnects which can reduce the communication that is overhead, and which will improve the data storage. Through the predictive models and also with effective power gating, the improved energy management system will be especially surrounds by gathering the energy, which may maximize the energy utilization. The capacity which expands also which will be easy to deploy is crucial for CGRA designs to which it is applied through the wide range of the applications which is ranging from the large scale of monitoring the systems to IoT devices. We can tell you that the strong steps which is used to improve the security and dependability might make sure that it is safe and trustworthy operation in the extreme edge conditions. Further researchers may develop the energy minimal dataflow computing and which is used to build more adaptable and also that is efficient for computing systems which is for the cutting edge applications which is done by tackling. [4]

VII. FUTUREWORK

Future research in this particular field, which is the energy minimal dataflow computation will be going to focus on the potential areas that enhance the capabilities and the efficiency of the CGRA architectures. There is some advanced improvement in the compiler optimizations, for the purpose of fully covering the potentials of the proposed architectures. There is also a space for the improvement in the thread management, parallelism and the allocation of the resources. There can be the usage of Dynamic reconfiguration and the adaption mechanisms which can further improve the real time adjustments in the work load and the environmental changes, which helps in the enhancements of the performance and the efficiency in the consumption of the energy. We can also make an integrations of the inclusion of non-volatile memory and advanced interconnects, which will help in the reduction of the communication overhead and it also makes path to improve the storage of the data. There can be also a enhancements in the security and the reliability through the measures, which can ensure a safe and dependent operations in the extreme edge computing. So these are the areas where the researchers can further advance in the field which we are doing the survey, which helps in the creation of more versatile and efficient computing systems for the extreme edge computing.

REFERENCES

[1] Thilini Kaushalya Bandara, Dhananjaya Wijerathne, Tulika Mitra, and Li-Shiuan Peh. 2022. REVAMP: A Systematic Framework for Heterogeneous CGRA Real ization. In Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (Lausanne, Switzer land) (ASPLOS 2022). Association for Computing Machinery, New York, NY, USA, 918–932. https://doi.org/10.1145/3503222.3507772

[2] Han Cai, Ji Lin, Yujun Lin, Zhijian Liu, Haotian Tang, Hanrui Wang, Ligeng Zhu, and Song Han. 2022. Enable Deep Learning on Mobile Devices: Methods, Systems, and Applications. ACM Trans. Des. Autom. Electron. Syst. 27, 3, Article 20 (mar 2022), 50 pages. https://doi.org/10.1145/3486618

[3] Vidushi Dadu, Jian Weng, Sihao Liu, and Tony Nowatzki. 2019. Towards general purpose acceleration by exploiting common data-dependence forms. In Proceed ings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture. 924–939.

[4] Harsh Desai, Matteo Nardello, Davide Brunelli, and Brandon Lucia. 2022. Ca maroptera: A Long-Range Image Sensor with Local Inference for Remote Sensing Applications. ACM Trans. Embed. Comput. Syst. 21, 3, Article 32 (may 2022), 25 pages. https://doi.org/10.1145/3510850

[5] Graham Gobieski, Ahmet Oguz Atli, Kenneth Mai, Brandon Lucia, and Nathan Beckmann. 2021. Snafu: an ultra-low-power, energy-minimal CGRA-generation framework and architecture. In 2021 ACM/IEEE 48th Annual International Sympo sium on Computer Architecture (ISCA). IEEE, 1027–1040.

[6] Graham Gobieski, Brandon Lucia, and Nathan Beckmann. 2019. Intelligence Beyond the Edge: Inference on Intermittent Embedded Systems. In ASPLOS.

[7] Graham Gobieski, Amolak Nagi, Nathan Serafin, Mehmet Meric Isgenc, Nathan Beckmann, and Brandon Lucia. 2019. MANIC: A Vector-Dataflow Architecture for Ultra-Low-Power Embedded Systems. In MICRO.

[8] Zhaoying Li, Dan Wu, Dhananjaya Wijerathne, and Tulika Mitra. 2022. LISA: Graph Neural Network based Portable Mapping on Spatial Accelerators. In 2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA). IEEE, 444–459.

[9] Mahim Mishra, Timothy J Callahan, Tiberiu Chelcea, Girish Venkataramani, Seth C Goldstein, and Mihai Budiu. 2006. Tartan: evaluating spatial computation for whole program execution. ACM SIGARCH Computer Architecture News 34, 5 (2006).

[10] G. Ansaloni, P. Bonzini, and L. Pozzi. 2011. EGRA: A Coarse Grained Reconfig urable Architectural Template. IEEE Transactions on Very Large Scale Integration (VLSI) Systems. https://doi.org/10.1109/TVLSI.2010.2044667

[11] Volker Baumgarten, G. Ehlers, Frank May, Armin Nückel, Martin Vorbach, and MarkusWeinhardt. 2003. PACTXPP—Aself-reconfigurable data processing archi tecture. The Journal of Supercomputing. https://doi.org/10.1023/A:102449960157

[12] Lattner Chris and Adve Vikram. 2004. LLVM: A Compilation Framework for Lifelong Program Analysis & Transformation. In Proceedings of the Interna tional Symposium on Code Generation and Optimization: Feedback-Directed and Runtime Optimization. https://doi.org/10.1109/CGO.2004.1281665

[13] M.R. Guthaus, J.S. Ringenberg, D. Ernst, T.M. Austin, T. Mudge, and R.B. Brown. 2001. MiBench: A free, commercially representative embedded benchmark suite. In Proceedings of the Fourth Annual IEEE International Workshop on Workload Characterization. WWC-4 (Cat. No.01EX538). https://doi.org/10.1109/WWC.2001. 990739

[14] C. Kim, M. Chung, Y. Cho, M. Konijnenburg, S. Ryu, and J. Kim. 2012. ULP SRP: Ultra low power Samsung Reconfigurable Processor for biomedical bookti tle=2012 International Conference on Field-Programmable Technology, applica tions. https://doi.org/10.1109/FPT.2012.6412157

[15] Leibo Liu, Jianfeng Zhu, Zhaoshi Li, Yanan Lu, Yangdong Deng, Jie Han, Shouyi Yin, and Shaojun Wei. 2019. A Survey of Coarse-Grained Reconfigurable Architecture and Design: Taxonomy, Challenges, and Applications. (2019). https://doi.org/10.1145/3357375

[16] Chris Nicol. 2017. A Coarse Grain Reconfigurable Array (CGRA) for Statically Scheduled Data Flow Computing.